DATABASE PROGRAMMING WITH PL/SQL
COURSE CODE: 5326

COURSE DESCRIPTION: This curriculum covers PL/SQL, a procedural language extension to SQL. Through an innovative project-based approach, students learn programming logic constructs such as variables, constants, conditional statements, and iterative controls. The course blends hands-on exercises, industry-type assessments, and project based learning experiences while leveraging the latest Oracle technologies. Students utilize an Oracle hosted, state-of-the-art lab environment to develop database programming skills using PL/SQL and Oracle Application Express. Students continue to improve skills including problem solving, teamwork, project management, and technical presentations that are used in a variety of industries and job roles.

OBJECTIVE: Given the necessary equipment, supplies, and facilities, the student will complete all of the following core standards successfully.

RECOMMENDED GRADE LEVELS: 10-12

COURSE CREDIT: 1 or 2 unit(s)

PREREQUISITE: Database Design & Programming with SQL (5324)

COMPUTER REQUIREMENT: one computer per student; Internet access; see Oracle Academy specifications for additional information

RECOMMENDED SOFTWARE: Adobe Acrobat Professional; Mozilla Firefox; Google Chrome

RESOURCES: Oracle iLearning online curriculum; Oracle Application Express; both provided by Oracle Academy

INDUSTRY CREDENTIALS/CERTIFICATIONS AVAILABLE:

Students may complete the final requirement for the Oracle PL/SQL Developer Certified Associate by visiting a certified testing facility and passing either Programming with PL/SQL Exam 1Z0-147 or Oracle Database 11g: Programming with PL/SQL Exam 1Z0-144. Discount coupons for exams may be available from Oracle Academy. Third-party, supplemental practice exams and materials are recommended, and may be available at discounted prices.

Certification requirements and pathways are subject to change and should be confirmed prior to exam registration.

A. SAFETY AND ETHICS

1. Identify major causes of work-related accidents in offices.
2. Describe the threat of viruses to a computer network, methods of avoiding attacks, and options in dealing with virus attacks.
3. Identify potential abuse and unethical uses of computers and networks.
4. Explain the consequences of illegal, social, and unethical uses of information technologies, e.g., piracy; illegal downloading; licensing infringement; and inappropriate uses of software, hardware, and mobile devices.
5. Differentiate between freeware, shareware, and public domain software copyrights.
6. Discuss computer crimes, terms of use, and legal issues such as copyright laws, fair use laws, and ethics pertaining to scanned and downloaded clip art images, photographs, documents, video, recorded sounds and music, trademarks, and other elements for use in Web publications.
7. Identify netiquette including the use of email, social networking, blogs, texting, and chatting.
8. Describe ethical practices in business professions such as safeguarding the confidentiality of business-related information.

B. EMPLOYABILITY SKILLS

1. Identify positive work practices, e.g., appropriate dress code for the workplace, personal grooming, punctuality, time management, organization.
2. Demonstrate positive interpersonal skills, e.g., communication, respect, and teamwork.

C. STUDENT ORGANIZATIONS

1. Explain how related student organizations are integral parts of career and technology education courses.
2. Explain the goals and objectives of related student organizations.
3. List opportunities available to students through participation in related student organization conferences/competitions, community service, philanthropy, and other activities.
4. Explain how participation in career and technology education student organizations can promote lifelong responsibility for community service and professional development.

D. PROCEDURAL LANGUAGE FUNDAMENTALS

1. Explain the benefits of PL/SQL.
2. List differences between PL/SQL and other programming languages.
3. Describe the structure of a PL/SQL block.

4. Identify the different types of PL/SQL blocks.
5. Identify PL/SQL programming environments.
6. Create an anonymous PL/SQL block that generates output.

E. DEFINING VARIABLES AND DATATYPES

1. List the uses of variables in PL/SQL.
2. Identify PL/SQL variable syntax.
3. Use variables in a PL/SQL block.
4. Assign new values to variables in PL/SQL.
5. Define the different types of lexical units available in PL/SQL.
6. Identify valid and invalid identifiers in PL/SQL.
7. Identify reserved words, delimiters, literals, and comments in PL/SQL.
8. Explain the need for datatypes.
9. Describe datatype categories.
10. List examples of scalar and composite datatypes.
11. Use scalar datatypes in a PL/SQL block.
12. Define guidelines for declaring and initializing PL/SQL variables.
13. Explain the benefits of anchoring data types with the %TYPE attribute.
14. Construct accurate variable assignment statements in PL/SQL.
15. Construct accurate statements using built-in SQL functions in PL/SQL.
16. Explain implicit and explicit conversions of datatypes.
17. List drawbacks of implicit datatype conversions.
18. Use functions to explicitly convert datatypes.
19. Construct statements using operators in PL/SQL.
20. Explain the scope and visibility of variables.
21. Identify variables using labels within nested blocks.
22. Explain the rules for variable scope in nested blocks.
23. Write PL/SQL code demonstrating good programming practices.
24. Write PL/SQL code with appropriate comments.
25. Write PL/SQL code using readability formatting guidelines.

F. USING SQL WITHIN PL/SQL

1. Create DML statements to insert, update, delete, and merge data.
2. Identify SQL statements that can be used in a PL/SQL block.
3. Create an INTO clause to hold the values returned by a single-row SQL SELECT statement.
4. Use good practice guidelines to write SQL SELECT statements.
5. Apply good practice guidelines for naming variables.
6. Create PL/SQL blocks that manipulate data with DML statements.
7. Describe when to use implicit or explicit cursors in PL/SQL.
8. Create PL/SQL code to use SQL implicit cursor attributes to evaluate cursor activity.
9. Define a transaction.
10. Create transaction control statements in PL/SQL.

G. CONDITIONAL CONTROL STRUCTURES

1. Define the types of conditional control structures.
2. Create an IF statement.
3. Create an IF-THEN-ELSIF-ELSE statement.
4. Create PL/SQL to handle the null condition in IF statements.
5. Use CASE statements and CASE expressions in PL/SQL.
6. Create CASE statements to handle null conditions in PL/SQL.
7. Create CASE and IF statements to handle Boolean conditions in PL/SQL.
8. Explain the different types of LOOP statements.
9. Create a basic loop with an EXIT statement.
10. Create a basic loop with an EXIT statement using conditional termination.
11. Explain when to use a FOR loop versus a WHILE loop.
12. Create a WHILE loop.
13. Create a FOR loop.
14. Create PL/SQL using nested loops.
15. Label loops and use the labels in EXIT statements.
16. Evaluate a nested loop construct and identify the exit point.

H. CURSORS AND PARAMETERS

1. Describe when to use an explicit cursor.
2. List guidelines for declaring and controlling explicit cursors.
3. Create code that opens a cursor, fetches a piece of data into a variable, and closes the cursor.
4. Use a simple loop to fetch multiple rows from a cursor.
5. Define a record structure using the %ROWTYPE attribute.
6. Create code to process the row of an active set using record types in cursors.
7. Create code to retrieve information about the state of an explicit cursor using cursor attributes.
8. Explain the benefits of using cursor FOR loops.
9. Create code to declare a cursor and manipulate it in a FOR loop.
10. Create code containing a cursor FOR loop using a subquery.
11. List the benefits of using parameters with cursors.
12. Create code to declare and manipulate a cursor with a parameter.
13. Create code to lock rows before an update.
14. Explain the effect of using NOWAIT in a update cursor declaration.
15. Create code to use the current row of the cursor in an UPDATE or DELETE statement.
16. Explain the use of multiple cursors to produce multilevel reports.
17. Create code using multiple cursors within nested loops.
18. Create code that manipulates multiple cursors using parameters.

I. COMPOSITE DATATYPES

1. Create user-defined PL/SQL records.
2. Manipulate user-defined PL/SQL records.
3. Create an INDEX BY table.
4. Create an INDEX BY table of records.
5. Describe the difference between records, tables, and tables of records.

J. EXCEPTION HANDLING

1. Identify advantages of using exception handling code.
2. Create PL/SQL code to include an EXCEPTION section.
3. List guidelines for exception handling.
4. Identify Oracle defined exceptions.
5. Identify user-defined exceptions.
6. Explain the difference between implicitly and explicitly handled errors.
7. Write code to trap a predefined Oracle error.
8. Write code to trap a non-predefined Oracle error.
9. Write code to identify an exception by error code and by error message.
10. Write code to name a user-defined exception.
11. Write code to raise an exception.
12. Write code to handle a raised exception.
13. Write code to use RAISE_APPLICATION_ERROR.
14. Describe the scope of an exception.
15. Describe the effect of exception propagation in nested blocks.

K. CREATING SUBPROGRAMS – PROCEDURES

1. Define anonymous blocks and subprograms.
2. Identify benefits of subprograms.
3. Define a stored procedure.
4. Create a procedure.
5. Describe how a stored procedure is invoked.
6. List the development steps for creating a procedure.
7. Create a nested subprogram in the declarative section of a procedure.
8. Describe how parameters contribute to a procedure.
9. Create a procedure using a parameter.
10. Invoke a procedure that has parameters.
11. Explain the difference between formal and actual parameters.
12. List the types of parameter modes.
13. Create a procedure that passes parameters.
14. Identify methods for passing parameters.
15. Describe the DEFAULT option for parameters.

L. CREATING SUBPROGRAMS – FUNCTIONS

1. Define a stored function.
2. Create a stored function.
3. List ways in which a function can be invoked.
4. Create code that invokes a function that has parameters.
5. List the development steps for creating a function.
6. Describe the differences between procedures and functions.
7. List the advantages of user-defined functions in SQL statements.
8. Explain when and where user-defined functions can be called.
9. Describe the purposes of the Data Dictionary.
10. Differentiate between the three types of Data Dictionary views.
11. Write SQL SELECT statements to retrieve information from the Data Dictionary.
12. Explain the use of DICTIONARY as a Data Dictionary search engine.
13. Describe how exceptions are propagated.
14. Remove a function and a procedure.
15. Use Data Dictionary views to identify and manage stored programs.

M. GROUPING SUBPROGRAMS USING PACKAGES

1. Describe the reasons for using a package.
2. Describe the specification and body components of a package.
3. Create packages containing related variables, cursors, constants, exceptions, procedures, and functions.
4. Create a PL/SQL block that invokes a package construct.
5. Explain the difference between public and private package constructs.
6. Identify a package construct as either public or private.
7. Identify the appropriate syntax to drop packages.
8. Identify views in the Data Dictionary used to manage packages.
9. Identify guidelines for using packages.
10. Create packages that use the overloading feature.
11. Create packages that use forward declarations.
12. Explain the purpose of a package initialization block.
13. Create a bodiless package.
14. Invoke packaged functions from SQL.
15. Identify restrictions on using packaged functions in SQL statements.
16. Create a package that uses PL/SQL tables and records.

N. EXPLORING ORACLE SUPPLIED PACKAGES

1. Identify persistent states of package variables.
2. Control the persistent state of a package cursor.
3. Describe uses for the DBMS_OUTPUT server-supplied package.
4. Identify the syntax to specify messages for the DBMS_OUTPUT package.
5. Describe the purpose for the UTL_FILE server-supplied package.

6. Describe the exceptions used in conjunction with the UTL_FILE server-supplied package.
7. Describe the main features of the UTL_MAIL server-supplied package.

O. DYNAMIC SQL AND PL/SQL PERFORMANCE

1. Identify the execution stages of SQL statements.
2. Describe the reasons for using dynamic SQL.
3. List four PL/SQL statements supporting Native Dynamic SQL.
4. Describe the benefits of EXECUTE IMMEDIATE over DBMS_SQL.
5. Identify the benefits of the NOCOPY hint.
6. Identify the benefits of the DETERMINISTIC clause.
7. Create subprograms that use the NOCOPY hint and the DETERMINISTIC clause.
8. Use Bulk Binding FORALL in a DML statement.
9. Use BULK COLLECT in a SELECT statement.
10. Use BULK COLLECT in a FETCH statement.
11. Use the Bulk Binding RETURNING clause.

P. USING AND MANAGING TRIGGERS

1. Describe database triggers.
2. Explain the difference between a database trigger and an application trigger.
3. List guidelines for using triggers.
4. Explain the differences between database triggers and stored procedures.
5. Explain the similarities of database triggers and stored procedures.
6. Create a DML trigger.
7. List the DML trigger components.
8. Create a statement level trigger.
9. Describe the trigger firing sequence options.
10. Create a DML trigger that uses conditional predicates.
11. Create a row-level trigger.
12. Create a row-level trigger that uses OLD and NEW qualifiers.
13. Create an INSTEAD OF trigger.
14. Create a Compound Trigger.
15. Describe events that cause DDL and database event triggers to fire.
16. Create a trigger for a DDL statement.
17. Create a trigger for a database event.
18. Describe the functionality of the CALL statement.
19. Describe the cause of a mutating table.
20. Write code to retrieve trigger information from the Data Dictionary.
21. Write code to disable a database trigger.
22. Write code to enable a database trigger.
23. Remove a trigger from the database.

Q. MANAGING OBJECT DEPENDENCIES

1. Describe the implications of procedural dependencies.
2. Contrast dependent objects and referenced objects.
3. Write code to retrieve dependency information from the data dictionary.
4. Use the UTLDTREE script to create the objects required to display dependencies.
5. Use the IDEPTREE and DEPTREE views to display dependencies.
6. Describe when automatic recompilation occurs.
7. List how to minimize dependency failures.
8. Describe remote dependencies.
9. List how remote dependencies are controlled.
10. Describe when a remote dependency is unsuccessfully recompiled.
11. Describe when a remote dependency is successfully recompiled.

R. USING THE PL/SQL COMPILER

1. Describe how PLSQL_CODE_TYPE can improve execution speed.
2. Describe how PLSQL_OPTIMIZE_LEVEL can improve execution speed.
3. Use USER_PLSQL_OBJECT_SETTINGS to see how a PL/SQL program was compiled.
4. Explain the difference between a warning and an error.
5. Identify the warning levels set by the PLSQL_WARNINGS parameter.
6. Set warning levels by calling the DBMS_WARNING server-supplied package from within a PL/SQL subprogram.
7. Describe the benefits of conditional compilation.
8. Create code to conditionally compile a PL/SQL subprogram containing selection, inquiry, and error directives.
9. Create code to conditionally compile a PL/SQL program which calls the DBMS_DB_VERSION server-supplied package.
10. Describe the benefits of obfuscated PL/SQL code.
11. Use the DBMS_DDL.CREATE_WRAPPED server-supplied procedure.
12. Describe how to use the Wrapper utility to obfuscate PL/SQL code.